

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200206306-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): John S. Worley

Confirmation No.: 5520

Application No.: 10/659,837

Examiner: Chuong D. Ngo

Filing Date: September 10, 2003

Group Art Unit: 2193

Title: METHOD AND SYSTEM FOR HIGH PERFORMANCE, MULTIPLE-PRECISION MULTIPLY-AND-ADD OPERATION

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on June 7, 2007.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month
\$120

☐ 2nd Month
\$450

☐ 3rd Month
\$1020

☐ 4th Month
\$1590

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of \$ 500 . At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees.

☒ A duplicate copy of this transmittal letter is enclosed.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:
Commissioner for Patents, Alexandria, VA 22313-1450
Date of Deposit: August 7, 2007

OR

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.

Date of facsimile:

Typed Name: Joanne Bourguignon

Signature:

Respectfully submitted,

John S. Worley

By

Robert W. Bergstrom

Attorney/Agent for Applicant(s)

Reg No. : 39,906

Date : August 7, 2007

Telephone : 206.621.1933



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicant: John S. Worley
Application No.: 10/659,837
Filed: September 10, 2003
Title: Method and system for high performance, multiple-precision
multiply-and-add operation

Examiner: Chuong D. Ngo

Art Unit: 2193

Docket No.: 200206306-1

Date : August 7, 2007

APPEAL BRIEF

Mail Stop: Appeal Briefs – Patents
Commissioner of Patents and Trademarks
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This appeal is from the decision of the Examiner, in an Office Action mailed February 7, 2007, finally rejecting claims 1-19.

08/13/2007 HDEMESS1 00000042 082025 10659837

01 FC:1402 500.00 DA

REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 20555 S.H. 249 Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

RELATED APPEALS AND INTERFERENCES

Appellant's representative has not identified, and does not know of, any other appeals of interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-19 were finally rejected in the Office Action dated February 7, 2007. Appellants' appeal the final rejection of claims 1-19 which are copied in the attached CLAIMS APPENDIX.

STATUS OF AMENDMENTS

No Amendment After Final is enclosed with this brief. The last Amendment was filed April 9, 2007.

SUMMARY OF CLAIMED SUBJECT MATTER

Overview

The present invention is directed to a computer operation, implemented as a series of processor instructions, that implements a multiple-precision, multiply-and-add operation. The computer operation is, as discussed in the current application, far more efficient than currently-available multiple-precision, multiply-and-add operations. Currently available multiple-precision, multiply-and-add operations, as discussed beginning on line 26 of page 10 of the current application, are implemented such that many write dependencies between instructions prevent effective use of parallel instruction execution. The multiple-precision-multiply-and-add-operation embodiments of the present invention eliminate write dependencies within instruction blocks, allowing for parallel execution of multiple instructions to provide far greater computational efficiency.

Independent Claim 1

Claim 1 is directed to a multiple-precision, multiply-and-add computer operation (page 13, line 27 to page 17, line 2) for multiplying together a first operand with a second operand, at least one of the first and second operands having more than one natural

word, and then adding an addend operand to the product of the first and second operands to produce a final result that is written into a multiple-natural-word-containing result vector. The multiple-precision, multiply-and-add operation comprises: (1) the first operand (406); (2) the second operand (408); (3) the addend operand (410); (4) the result vector (412); and (5) for each natural word of the second operand, a block of *multiply-and-add* instructions (page 14, lines 5-39) that multiply the natural word of the second operand by all natural words of the first operand and store results of the multiply-and-add instructions as intermediate results, the block of *multiply-and-add* instructions that multiply the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

Dependent Claims 2-9

Claim 2 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein each block of *multiply-and-add* instructions (page 14, lines 5-39) contains only *multiply-and-add* instructions. Claim 3 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein a block of *multiply-and-add* instructions may contain *add* instructions in addition to *multiply-and-add* instructions (page 17, lines 23-25). Claim 4 is directed to the multiple-precision, multiply-and-add operation of claim 1 further including a number of blocks of *add* instructions (page 14, line 41 to page 15, line 23) that add the intermediate results and any remaining natural words of the addend operand to produce a final result that is stored into the result vector. Claim 5 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein at least one of the first operand, second operand, and addend operand is contained within two or more registers (page 18, lines 22-25). Claim 6 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein at least one of the first operand, second operand, and addend operand is contained within two or more natural words in memory (page 18, lines 22-25). Claim 7 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein the result vector is contained within two or more registers (412). Claim 8 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein the result vector is contained within two or more natural words in memory (page 18, lines 22-25). Claim 9 is directed to the multiple-precision, multiply-and-add operation of claim 1 wherein, because there are no write

dependencies in the blocks of *multiply-and-add* instructions, all *multiply-and-add* instructions of each block (page 14, lines 5-39) can be executed together in parallel (page 17, lines 7-8).

Independent Claim 10

Claim 10 is directed to a method, carried out by a computer, for multiplying a first operand (406) by a second operand (408) to produce an intermediate product (414) to which an addend operand (410) is added to produce a result stored in a result vector (412), at least one of the first operand, second operand, and addend operand having more than one natural word (page 13, line 27 to page 17, line 2). The method comprises, for each natural word of the second operand, using a block of *multiply-and-add* instructions (page 14, lines 5-39) to multiply the natural word of the second operand (408) by all natural words of the first operand (406) and store results of the *multiply-and-add* instructions as intermediate results (414), when multiplying the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

Dependent Claims 11-18

Claim 11 is directed to the method of claim 10 wherein each block of *multiply-and-add* instructions (page 14, lines 5-39) contains only *multiply-and-add* instructions. Claim 12 is directed to the method of claim 10 wherein a block of *multiply-and-add* instructions may contain *add* instructions in addition to *multiply-and-add* instructions (page 17, lines 23-25). Claim 13 is directed to the method of claim 10 further including using a number of blocks of *add* instructions (page 14, line 41 to page 15, line 23) that add the intermediate results and any remaining natural words of the addend operand to produce a final result that is stored into the result vector that contains a sum of the addend operand and a product of the first and second operands. Claim 14 is directed to the method of claim 10 wherein at least one of the first operand, second operand, and addend operand is contained within two or more registers (page 18, lines 22-25). Claim 15 is directed to the method of claim 10 wherein at least one of the first operand, second operand, and addend operand is contained within two or more natural words in memory (page 18, lines 22-25). Claim 16 is directed to the method of claim 10 wherein the result vector is contained within two or more

registers (412). Claim 17 is directed to the method of claim 10 wherein the result vector is contained within two or more natural words in memory (page 18, lines 22-25). Claim 18 is directed to the method of claim 10 further including executing some or all of the *multiply-and-add* instructions of each block of *multiply-and-add* instructions (page 14, lines 5-39) in parallel (page 17, lines 7-8).

Independent Claim 19

Claim 19 is directed to a multiple-precision, multiply-and-add computer operation (page 13, line 27 to page 17, line 2) for multiplying together a first operand with a second operand, at least one of the first and second operands having more than one natural word, and then adding an addend operand to the product of the first and second operands to produce a final result that is written into a multiple-natural-word-containing result vector, the multiple-precision. The multiply-and-add operation comprises: (1) the first operand (406); (2) the second operand (408); (3) the addend operand (410); (4) the result vector (412); and (5) for each natural word of the second operand, a means for multiplying the natural word of the second operand by all natural words of the first operand (page 14, lines 5-39) and storing results as intermediate results, the means for multiplying the natural word of the second operand by all natural words of the first operand additionally adds a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand without write dependencies.

GROUND OF REJECTION TO BE REVIEWED ON APPEAL

1. The rejection of claims 1-19 under 35 U.S.C. § 101 for being directed to non-statutory subject matter.

ARGUMENT

Claims 1-19 are pending in the current application. In an Office Action dated February 7, 2007 ("Office Action"), the Examiner rejected claims 1-19 under 35 U.S.C. § 101. In an advisory action dated April 27, 2007, the Examiner maintained the 35 U.S.C. § 101 rejections of claims 1-19. Appellant respectfully traverses these rejections.

ISSUE 1**The rejection of claims 1-19 under 35 U.S.C. § 101 for being directed to non-statutory subject matter.**

In rejecting claims 1-19 under 35 U.S.C. §101, the Examiner states, in the Office Action:

As per claims 1-19, the invention is a computer implemented method of calculation. In order for a claimed invention that is directed to such a computer implemented method to be statutory, the claimed invention must accomplish a practical application. That is the claimed invention must transform an article or physical object to a different state or thing, or produce a useful, concrete and tangible result. State Street, 149 F.3d at 1373-74, 47 USPQ2d at 1601-02. Also see "Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility", OG Notices: 22 November 2005. It is clear from claims 10-18 that the claims merely involve calculations and manipulations of data in performing calculations. The claimed invention does not transform an article or physical object to a different state or thing. The inputs are number and the output are also numbers. The result of the invention is merely numerical values without a practical application recited in the claims to make the result useful, concrete and tangible. Therefore, the claimed invention is directed to non-statutory subject matter as the claims fail to assert a practical application to the invention.

It is respectfully submitted that the claimed invention is clearly directed to a calculation in a computer, namely a multiply-and-add operation. Such an invention that is directed to a calculation is a 35 U.S.C § 101 judicial exception as being an Abstract Idea, and is required a practical application of the judicial exception recited in the claimed in order to be statutory. See MPEP 2105, IV, C, 2. That is the claimed invention must transform an article or physical object to a different state or thing, or produce a useful, concrete and tangible result. Since there is no practical application of the multiply-and-add operation recited in the claim, the claimed invention is directed to non statutory subject matter. Further, the examiner respectfully submits that the greater efficiency and elimination of write dependence as disclosed in the specification are improvements of the invention in performing a calculation. They are not practical applications of the invention. The focus of determination whether the claimed invention produce a useful, concrete, and tangible result is not on whether the steps taken to achieve a particular result are useful, tangible, and concrete, but rather on whether the final result achieved by the claimed invention is useful, tangible and concrete. Therefore, it is respectfully submitted that the rejection of claims 1-19 under 35 U.S.C. § 101 is proper.

In reiterating the rejection of claims 1-19 in the Advisory Action, the Examiner states:

Continuation of 11. does NOT place the application in condition for allowance because: applicant's arguments are not persuasive to overcome the rejection since the result produced by the claimed invention is a mere numerical value indicating a multiply-and-add operation result without a practical application to the result to make it useful. Further claims 1-9 and 19 are claiming operation which clearly cover a computer program per se.

Appellant's representative does not completely follow the logic of the Examiner's statements, quoted above. The thrust of the rejection appears to be that the Examiner does not regard computer programs as patentable subject matter. Appellant's

representative knows of no *per se* prohibition with regard to the patenting of software. The recently promulgated "Interim Guidelines for Examination of Patent Applications for Patent Subject Matter Eligibility" ("Guidelines"), issued by the USPTO, provides no basis for such a statement. 35 U.S.C. §101 contains neither an explicit prohibition against patenting software nor suggestion that software is *per se* unpatentable. As quoted in the Guidelines, the Federal Circuit has stated that:

Thus, it is improper to read into §101 limitations as to the subject matter that may be patented where the legislative history does not indicate that Congress clearly intended such limitations. *Alappat*, 33 F.3d 1542, 31 USPQ2d 1556.

Furthermore, the Guidelines quote the Supreme Court holding that: "Congress chose expansive language of 35 U.S.C. §101 as to include 'anything under the sun that is made by man.'" The Guidelines further quote Diamond v. Chakrabarty, 447 U.S. 303, 308-09, 206 USPQ 193, 197 (1980) as stating:

In choosing such expansive terms as "manufactured" and "composition of matter," modified by the comprehensive "any," Congress plainly contemplated that the patent laws would be given wide scope. The relevant legislative history also supports a broad construction.

The Examiner's justification for rejecting claims 9-16 under 35 U.S.C. §101 that, according to the Examiner, software is *per se* unpatentable subject matter, would appear to quite clearly contradict the highest authorities on the subject of subject-matter patentability. Moreover, the USPTO has issued many thousands of patents directed to software inventions, including dozens of software patents prepared and prosecuted by Appellant's representative. The Examiner's statement not only appears to be unfounded, but, on view of the many issued software-related patents, quite unreasonable.

A second thrust of the Examiner's arguments appears to be that the Examiner considers the currently claimed invention to be an abstract idea without a practical application. The Examiner several times states that the claimed invention does not transform an article or physical object to a different state or thing, and several times states that the currently claimed multiply-and-add operation has no practical application.

First, the currently claimed invention is not an abstract idea. A very well-documented and complete pseudo-code implementation of one embodiment of the claimed multiple-position multiply-and-add operation is provided in the current application beginning on line 29 of page 5 of the current application. A computer program or routine that carries

out a well-defined operation is not an abstract idea. As is well known to those familiar to computer science and computer hardware, a computer program or routine executing within a general-purpose computer essentially transforms the general-purpose computer into a special-purpose computer, or machine, that carries out the task encoded in the computer program or routine. As is well known to those familiar with computer science and computer hardware, such programs or routines encoded as sets of instructions can be implemented in either software, on a general purpose computer, or in hardware circuits. The currently claimed invention is not directed to a mathematical algorithm, abstract idea, natural phenomenon, or law of nature. Instead, it is directed to a carefully encoded, well-characterized, deterministic set of logical operations that carry out a well-defined task within a computer system, essentially comprising a component of the computer system that is used for carrying out higher-level tasks, including cryptography-related tasks.

According to the Guidelines, "[t]he applicant is in the best position to explain why an invention is believed useful." Beginning on line 22 of page 3 of the current application, Appellant discusses one useful application of the multiple-precision multiply-and-add operation to which the current claims are directed. As is well known to those familiar with computer science, multiple-precision multiply-and-add operations are fundamental to many modern cryptographic methodologies, including methodologies that employ cryptographic hashes and a familiar public-key/private-key encryption methods. As discussed in the current application beginning on line 16 of page 10, current implementations of multiple-precision multiply-and-add operations are quite inefficient, in that there are there are multiple write dependencies between instructions, limiting the degree to which parallel execution of the instructions can be used to increase the performance of the multiple-precision multiply-and-add operations. Similar write-dependency problems are discussed, again, beginning on line 16 of page 13 of the current application. Finally, embodiments of the present invention are discussed, beginning on line 27 of page 13 of the current application. In these embodiments, write dependencies are avoided, allowing for full parallelization of the multiple-precision multiply-and-add operations, leading to significantly increased computational efficiency on modern processors, as explained in the current application. Appellant has clearly laid out reasons why the current invention is both needed and useful. As can be well appreciated by those familiar with computer science and computer technology, a large fraction of the innovations in computer-related fields during the past 50 years are directed to increasing the speed of operation of computer systems and

components of computer systems.

According to the Guidelines, there are a number of judicial exceptions to the patentable subject matter defined in 35 U.S.C. §101, the judicial exceptions including: (1) laws of nature; (2) natural phenomena; and (3) abstract ideas. Neither a "multiple-precision, multiply-and-add instruction" nor a "method, carried out by a computer, for multiplying a first operand by a second operand to produce an intermediate product to which an addend operand is added to produce a result stored in a result vector" falls anywhere close to these judicial exceptions. In fact, a multiple-precision, multiply-and-add instruction is a machine component that serves a vital function in many modern computer systems. Whether the multiple-precision, multiply-and-add instruction is implemented exclusively in software, in a combination of software or hardware, exclusively in hardware, or in a combination of firmware and hardware appears to be quite immaterial to its patentability under 35 U.S.C. §101, case law, and the Guidelines.

According to the Guidelines, even in the case of the judicial exceptions, *practical applications* of abstract ideas, laws of nature, or natural phenomena may indeed be patentable. Various tests are provided in the Guidelines. For example, as stated in the Guidelines, if the claimed subject matter "transforms" an article or physical object to a different state or thing, it is patentable. Please consider the three independent claims, provided below, with added emphasis to point out claim language clearly directed to transformation of physical objects, namely electronic memory and/or registers, to a different state:

1. A multiple-precision, multiply-and-add ***computer operation*** for multiplying together a first operand with a second operand, at least one of the first and second operands having more than one natural word, and then adding an addend operand to the product of the first and second operands ***to produce a final result that is written into a multiple-natural-word-containing result vector***, the multiple-precision, multiply-and-add operation comprising:

- the first operand;
- the second operand;
- the addend operand;
- the result vector; and
- for each natural word of the second operand,

- a block of *multiply-and-add* instructions that multiply the natural word of the second operand by all natural words of the first operand ***and store results of the multiply-and-add instructions as intermediate results***, the block of *multiply-and-add* instructions that multiply the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

10. A method, *carried out by a computer*, for multiplying a first operand by a second operand *to produce an intermediate product* to which an addend operand is added *to produce a result stored in a result vector*, at least one of the first operand, second operand, and addend operand having more than one natural word, the method comprising:

for each natural word of the second operand,

using a block of *multiply-and-add* instructions to multiply the natural word of the second operand by all natural words of the first operand *and store results of the multiply-and-add instructions as intermediate results*, when multiplying the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

19. A multiple-precision, multiply-and-add *computer operation* for multiplying together a first operand with a second operand, at least one of the first and second operands having more than one natural word, and then adding an addend operand to the product of the first and second operands *to produce a final result that is written into a multiple-natural-word-containing result vector*, the multiple-precision, multiply-and-add operation comprising:

the first operand;

the second operand;

the addend operand;

the result vector; and

for each natural word of the second operand,

a means for multiplying the natural word of the second operand by all natural words of the first operand *and storing results as intermediate results*, the means for multiplying the natural word of the second operand by all natural words of the first operand additionally adds a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand without write dependencies.

As anyone even cursorily familiar with computer systems or computer hardware and software well understands, storing the result of a multiply-and-add instruction within a computer system necessarily involves a transformation of the memory and/or registers used to contain the result of the multiply-and-add instruction. If there is no physical transformation, then the result of multiply-and-add instructions cannot be stored. Writing a result to memory and/or to registers necessarily involves changing the state of computer memory and/or computer registers. The state change generally involves changes in voltages within electrical components that represent binary bits, with a first voltage representing binary value "0" and a second voltage representing binary value "1." Moreover, Figures 4A-K illustrate, in great detail, the many step-by-step changes to memory and/or registers within a computer system as the multiple-precision, multiply-and-add computer operation to which the current claims are directed is carried out. Thus, the Examiner is quite incorrect in stating that the currently claimed invention does not transform an article or physical object to a different state or thing.

According to the Guidelines:

The Examiner first shall review the claim and determine if it provides a transformation or reduction of an article to a different state or thing. If the Examiner finds such a transformation or reduction, the Examiner shall end the inquiry and find that the claim meets the statutory requirement of 35 U.S.C. §101.

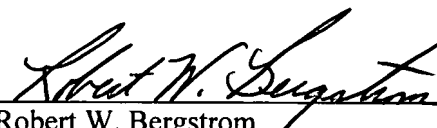
Thus, according to the Guidelines, the fact that the currently claimed invention, implemented as a computer operation, stores results of the multiply-and-add instructions as intermediate results and stores a result of the multiple-precision, multiply-and-add operation in a result vector necessarily implies that the currently claimed invention falls into the practical-application exception, and is patentable subject matter.

CONCLUSION

The current claims are directed to a multiple-precision, multiply-and-add computer operation and a method for carrying out a multiple-precision, multiply-and-add operation in a computer. Appellant has quite clearly described the need for, and practical utility of the current invention, and has thoroughly described the current invention in detailed pseudo-code and step-by-step illustrations of the contents of computer memory and computer registers (Figures 4A-4K) as the multiple-precision, multiply-and-add computer operation executes. The currently claimed invention clearly claims many physical transformations, namely the storing and reading of data in computer memory and/or computer registers. There is no basis in either 35 U.S.C. § 101, Federal Circuit opinions, or the Guidelines for a subject-matter rejection of the current claims.

Appellant respectfully submits that all statutory requirements are met and that the present application is allowable over all the references of record. Therefore, Appellant respectfully requests that the present application be passed to issue.

Respectfully submitted,
John S. Worley
OLYMPIC PATENT WORKS PLLC

By 
Robert W. Bergstrom
Registration No. 39,906

Olympic Patent Works ^{PLLC}
P.O. Box 4277
Seattle, WA 98104
206.621.1933 telephone
206.621.5302 fax

CLAIMS APPENDIX

1. A multiple-precision, multiply-and-add computer operation for multiplying together a first operand with a second operand, at least one of the first and second operands having more than one natural word, and then adding an addend operand to the product of the first and second operands to produce a final result that is written into a multiple-natural-word-containing result vector, the multiple-precision, multiply-and-add operation comprising:

the first operand;

the second operand;

the addend operand;

the result vector; and

for each natural word of the second operand,

a block of *multiply-and-add* instructions that multiply the natural word of the second operand by all natural words of the first operand and store results of the multiply-and-add instructions as intermediate results, the block of *multiply-and-add* instructions that multiply the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

2. The multiple-precision, multiply-and-add operation of claim 1 wherein each block of *multiply-and-add* instructions contains only *multiply-and-add* instructions.

3. The multiple-precision, multiply-and-add operation of claim 1 wherein a block of *multiply-and-add* instructions may contain *add* instructions in addition to *multiply-and-add* instructions.

4. The multiple-precision, multiply-and-add operation of claim 1 further including:

a number of blocks of *add* instructions that add the intermediate results and any remaining natural words of the addend operand to produce a final result that is stored into the result vector.

5. The multiple-precision, multiply-and-add operation of claim 1 wherein at least one of

the first operand, second operand, and addend operand is contained within two or more registers.

6. The multiple-precision, multiply-and-add operation of claim 1 wherein at least one of the first operand, second operand, and addend operand is contained within two or more natural words in memory.

7. The multiple-precision, multiply-and-add operation of claim 1 wherein the result vector is contained within two or more registers.

8. The multiple-precision, multiply-and-add operation of claim 1 wherein the result vector is contained within two or more natural words in memory.

9. The multiple-precision, multiply-and-add operation of claim 1 wherein, because there are no write dependencies in the blocks of *multiply-and-add* instructions, all *multiply-and-add* instructions of each block can be executed together in parallel.

10. A method, carried out by a computer, for multiplying a first operand by a second operand to produce an intermediate product to which an addend operand is added to produce a result stored in a result vector, at least one of the first operand, second operand, and addend operand having more than one natural word, the method comprising:

for each natural word of the second operand,

using a block of *multiply-and-add* instructions to multiply the natural word of the second operand by all natural words of the first operand and store results of the *multiply-and-add* instructions as intermediate results, when multiplying the first natural word of the second operand by all natural words of the first operand additionally adding a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand, the block of *multiply-and-add* instructions containing no write dependencies.

11. The method of claim 10 wherein each block of *multiply-and-add* instructions contains only *multiply-and-add* instructions.

12. The method of claim 10 wherein a block of *multiply-and-add* instructions may contain *add* instructions in addition to *multiply-and-add* instructions.

13. The method of claim 10 further including:

using a number of blocks of *add* instructions that add the intermediate results and any remaining natural words of the addend operand to produce a final result that is stored into the result vector that contains a sum of the addend operand and a product of the first and second operands.

14. The method of claim 10 wherein at least one of the first operand, second operand, and addend operand is contained within two or more registers.

15. The method of claim 10 wherein at least one of the first operand, second operand, and addend operand is contained within two or more natural words in memory.

16. The method of claim 10 wherein the result vector is contained within two or more registers.

17. The method of claim 10 wherein the result vector is contained within two or more natural words in memory.

18. The method of claim 10 further including executing some or all of the *multiply-and-add* instructions of each block of *multiply-and-add* instructions in parallel.

19. A multiple-precision, multiply-and-add computer operation for multiplying together a first operand with a second operand, at least one of the first and second operands having more than one natural word, and then adding an addend operand to the product of the first and second operands to produce a final result that is written into a multiple-natural-word-containing result vector, the multiple-precision, multiply-and-add operation comprising:

- the first operand;
- the second operand;
- the addend operand;
- the result vector; and

for each natural word of the second operand,

a means for multiplying the natural word of the second operand by all natural words of the first operand and storing results as intermediate results, the means for multiplying the natural word of the second operand by all natural words of the first operand additionally adds a number of initial natural words of the addend operand to the products of the first natural word of the second operand and all natural words of the first operand without write dependencies.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.